AD-A050 577 STANFORD UNIV CALIF STANFORD ELECTRONICS LABS F/G 9/2
EMMY/360 FUNCTIONAL CHARACTERISTICS.(U)
JUN 76 W A WALLACH
UNCLASSIFIED SU-SEL 76-024 ARO-12958.10-M DAAG-29-76-G-0001 NL

| OF |
AD A050577

END
DATE FILMED
4 -78
DDC

SEL-76-024

EMMY/360 Functional Characteristics

(3)

by

Walter A. Wallach

June 1976

Technical Report No. 114

The work described herein was supported in part by the Army Research Office-Durham under Grant DAAG-29-76-G-0001.

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY • STANFORD, CALIFORNIA

(14) SU-SEL 76-024,
DSL-TR-114

(6) EMMY/360 FUNCTIONAL CHARACTERISTICS.

by

(10) Walter A. Wallach

(11) June 1976

(12) 30 P.

(9) Technical Report, No. 114

(15) DAAG-29-76-G-0001

(78) ARO

(19) 12958.10-M

DIGITAL SYSTEMS LABORATORY
Stanford Electronics Laboratories
Stanford University
Stanford, CA 94305

The work described herein was supported in part by the Army Research Office-Durham under Grant DAAG-29-76-G-0001.

Digital Systems Laboratory

Stanford Electronics Laboratories

Technical Report No. 114

June 1976

EMMY/360 FUNCTIONAL CHARACTERISTICS

by

Walter A. Wallach

Abstract

An emulation of the IBM System/360 architecture is presented-the EMMY/360. Problem state code which executes correctly on an IBM 360 will also execute correctly on the EMMY/360. Code producing execution exceptions will, in most cases, produce the same results on the two systems. Certain exceptions occurring on IBM 360 cannot occur on the EMMY/360, such as address specification exceptions for main store operands, and certain precise interrupts on IBM 360 will be imprecise on the EMMY/360, such as address exceptions. The EMMY/360 supports the Standard 360 instruction set with single precision floating point. The 360 input/output structure is not supported; I/O on the EMMY system is done by Function Call instruction, rather than channel program and Start-Test I/O.

The work herein was supported in part by the Army Research Office-Durham under grant DAAG-29-76-G-0001.

1.0 Introduction

The EMMY/360 is a class B [5] emulator for the IBM System/360 architecture [2] written for the Stanford EMMY [3]. The current version supports the basic 360 instruction set (problem-state) with single precision floating point. The 360 input/output structure is not supported.

The EMMY/360 is intended to run the Stanford PL360 system developed by Wirth [9]. This system provides a single job monitor environment for the execution of PL360 programs. A 360 object text loader will be added to allow the execution of standard IBM problem state code, such as Fortran object, for the purpose of architectural evaluation.

The basic configuration of the EMMY/360 consists of the EMMY CPU [3] and the Datapoint 2200 terminal [1], which is used as a low speed I/O channel and diagnostic console. The Datapoint provides two 2400 baud cassette drives and access to a printer-keyboard, CRT-keyboard, paper tape reader-punch, and communications adapter.

System expansion plans call for the addition of a DEC PDP-11 ...ibus system [7] to support mass storage facilities, higher speed unit record equipment, and a more powerful channel/diagnostic console.

2.0 Compatability to IBM 360

The EMMY/360 maintains compatability with the IBM System/360 in problem state for the 360 basic instruction set. "Correct" code will produce the same results on both systems. Most exceptional conditions are handled the same in the emulator as in IBM 360. The differences will be discussed.

2.1 Pocessor State and the PSW

The emulator maintains a 64 bit PSW (decoded internally), which is returned to the progam in 360 basic control format at the appropriate main store address upon interruption (see [2]).

The emulator progam consists of 3 phases- operation decode (DECODE), operation execution (EXEC), and channel emulation (I/O). DECODE and EXEC proceed sequentially until an I/O operation or interruption is requested. The emulator then enters channel mode to service the transfer. In the case of an I/O request, the operands of the I/O instruction are checked, and a request to the channel processor (either the Datapoint or the PDP-11) is formatted. The channel processor then performs the transfer while the emulator returns control to the 360 program.

Channel completion or error is signalled by a hardware interrupt of the EMMY CPU. The channel emulator determines if the interrupt is allowed, and, if so, causes a trap to be taken to the PSW swap code the next time DECODE is entered. This insures the current 360 instruction will complete before the interrupt is serviced. If the interrupt is not allowed (masked out in the current PSW), the channel completion information is stored and, the next time the channel is accessed, a channel status word is returned to the program. The I/O request is not initiated in this case.

2.2 Addressing

2.2.1 Address Boundary Alignment

System 360 normally requires that operands be aligned on certain address boundaries. Full word (32 bit) operands must reside on byte addresses divisible by 4, and so forth. This restriction has been removed in the EMMY/360. All boundary alignment, multi word fetching, and buffering is performed by the EMMY main memory controller [4]. A byte, halfword or word operand

requested on an arbitrary byte address is always returned correctly by the memory controller. Unaligned operand fetch or store operations require two mainstore acceses; properly aligned operands will be accessed about twice as fast as unaligned ones.

2.2.2 Addressing Exception

All 360 addresses are treated as 24 bit unsigned integer values. No check is done as to the legality of requested 360 addresses. When a non-existant byte of main store is addressed, a Bus Timeout interrupt of the CPU will result (see [3]). This will be passed on to the 360 program as an imprecise addressing exception (ILC=0) if the request was made by the CPU. Note that, since the CPU cannot clear the bus following a timeout, the Datapoint or PDP-11 must clear the bus before the program can continue. Otherwise, the system will halt.

2.2.3 Storage Protect

No storage protect feature is provided. Any protection-related operation will result in an operation exception interrupt. The key field of the PSW is ignored.

2.3 Optional Features

2.3.1 Decimal Feature

Decimal feature is not supported. Any decimal feature instruction will result in an operation exception interruption. Decimal feature will be supported at a later date.

2.3.2 Dynamic Address Translation

At present, dynamic address translation will not be provided. In the future some form of virtual addressing will be supported. The 360-370 scheme of segment and page tables in main store will not be used, since this requires additional hardware such as CAM buffers.

A scheme based on a translation table in control store and paging control bits has been explored and will be implemented when the mass storage system becomes operational [8].

2.3.3 ASCII Mode

The EMMY/360 supports only EBCDIC mode (as in system/370). The ASCII bit of the PSW is ignored. All zone and sign digits generated are EBCDIC and all sign codes are assumed EBCDIC.

2.3.4 Floating Point

Only limited floating point support is provided. This includes the 360 single precision instructions of the 360 Floating Point Feature.

2.3.5 Interval Timer

An Interval Timer feature is included, which increments main store location 50 (hex) by one in the least significant bit position every 40,000 CPU cycles. When an overflow condition is detected (in the timer update operation), an EXTERNAL interruption of the program is generated with interrupt code X'80' (if not masked by PSW bit 56). The timer may be loaded with any value and read at any time. Resolution is 2.4 ms at 60ns internal cycle, and 1.4 ms at 35ns internal cycle.

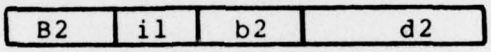
The timer may be used for event timing, time of day, or internal cycle counting.

3.0 I/O Support

3.1 Processor Support

The EMMY/360 does not support 360 channel program I/O. I/O requests are made by means of a Function Call instruction (OpCode = B2).

Input Output Function Call



i1 I/O operation requested

00	Read
02	Write
08	Rewind
09	Write Tape Mark
0A	Foreward Space TM
0B	Foreward Space Record
0C	Back Space TM
0D	Back Space Record

b2,d2 Unit Address

General Register 0 specifies Buffer Address
General Register 1 specifies Buffer Length

Condition Code

00	Operation Started
01	CSW Stored
10	Channel/Device Busy
11	Device Not Operational

Interrupt Action
Privileged Operation Exception

3.2 Program Support

The primary program support for the EMMY/360 is the PL360 Monitor System. It consists of a single job monitor, I/O routines, interrupt response and error recovery routines.

Progam level I/O requests are made via Supervisor Call instructions (SVC). Parameters are passed in General Registers 0, 1, and 2. The convention is as follows:

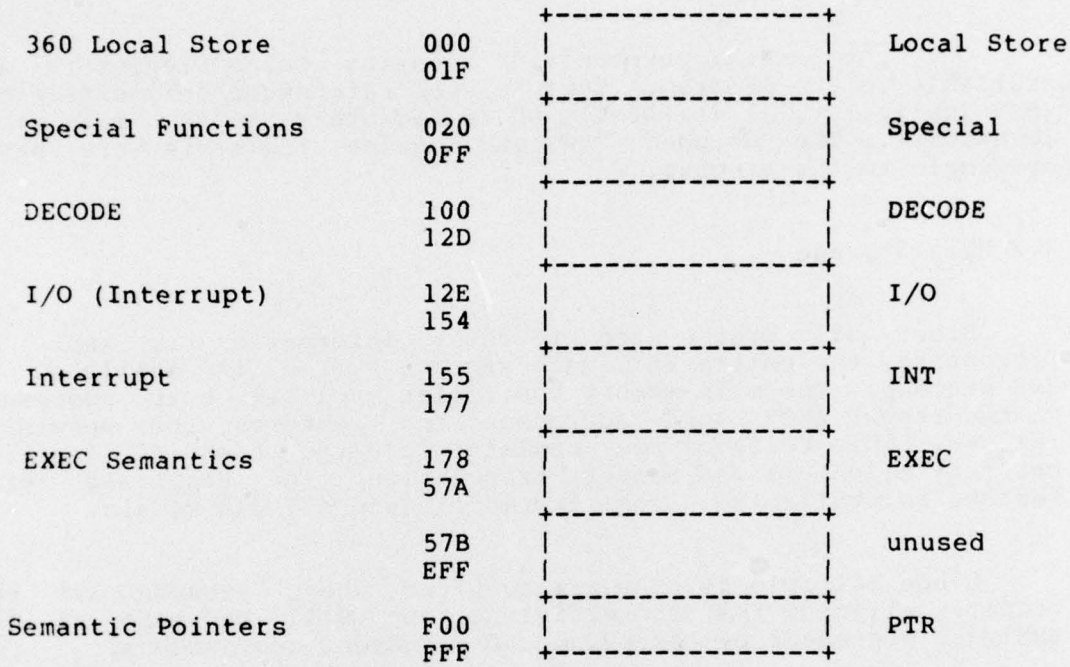
R0	contains buffer address
R1	contains buffer length
R2	contains Unit address

R1 may be modified by the I/O routine if the requested buffer length differs from the actual record read. The SVC routines map logical unit requests into device requests via a device table and code translation table, perform logical record buffering and code translation (unit record devices), and data transfer between system buffers and program buffers. Since the Monitor is a single job system, the program is forced to wait until the requested record has been transferred to its buffer. The condition code returned to the program reflects the outcome of the I/O operation (see [9] for details of program level I/O).

The monitor performs all tasks related to physical device access. The program merely issues an SVC instruction with the proper code and parameters. The monitor retrieves the device type and characteristics from the device table and issues the proper I/O Function Call. Note that, if the I/O request can be satisfied by a system buffer, data is transferred between this buffer and the user buffer and no I/O Function will be issued.

4.0 Emulator Program Logic

The 360 emulator for the EMMY/360 consists of 3 states- operation decode (DECODE), semantic execute (EXEC), and channel emulator (I/O). Microstore is organized in essentially 7 regions:



The regions are organized according to function into 5 segments. The low 36 words contain 360 local storage; general purpose registers, floating point registers, channel status registers, PSW, and so forth. The next segment contains special code such as IPL, PSW Restart, microcode to clear local and main storage. A third segment is comprised of DECODE and EXEC code, along with INT, the interrupt generating microcode. The fourth segment is channel emulator (I/O), and the last contains the semantic pointers.

4.1 Local Storage

The 360 architecture provides the programmer with 16 32-bit general purpose registers and 4 64-bit floating point registers.

These are stored in the low 20 words of microstore. The 360 architecture provides a 64 bit Progam Status Word which reflects the state of the 360 following an interruption. This is stored in microstore in decoded form. Part of the interrupt-generating code formats this information into a 360 basic control PSW (as opposed to extended control format used in the Model 67 and System/370 processors). Status and device registers for external, multiplexor and selector channel interrupt classes complete the emulated local storage.

Only the general purpose and floating point registers are available to the program. These may be referenced only explicitly (not implicitly, as in PDP-11, where registers are also main store locations). The decoded PSW and device registers are never available to the program.

4.2 Main Storage

Since all status and register information is kept in microstore, the entire EMMY main storage system is available as 360 storage. The main memory controller maps all byte addresses to the proper EMMY word addresses and performs the necessary fetches. This relieves the emulator of the task of checking boundary alignment and address translation. The unaligned fetch feature is available on some 360 models and all 370 models.

Since multiple fetches are required when operands are not properly aligned, the most efficient use will be made of the EMMY/360 processor by observing 360 alignment conventions.

4.3 Addressing

All local storage addresses are 4-bit addresses. General purpose registers, used for data storage and address completion, are mapped directly into the low 16 words of microstore. Floating point register addresses are also 4-bit addresses and are mapped into words 16 through 24 of microstore. All local storage addresses must conform to 360 address restrictions (for example, even/odd register pairs, double length register operands, and floating point register addresses always being even). A specification exception will result from any improper local storage address.

All main storage addresses are 24 bit unsigned values. No checking is done as to validity of addresses, and no alignment restrictions are enforced. Invalid addresses will result in a bus

timeout. The CPU will generate an imprecise (ILC=0) address exception interrupt if the timeout resulted from a CPU initiated operation and the bus is cleared by the control console processor (either the Datapoint or the PDP-11). Other wise the condition is ignored by the CPU.

When timeout code is entered, the CPU will halt. The processor which clears the bus must start the CPU, at which time EMMY checks if the last bus operation was CPU initiated (if so, the busy bit of Old Processor State Register will be 1). If the busy bit was zero, processing resumes. Otherwise, a program interrupt is taken with address exception specified. The ILC is set to 0.

5.0 Special Functions

The emulator provides certain special functions which simplify some operations.

5.1 Diagnostic Logout

When an error occurs during the emulation, or when requested by an external processor, a diagnostic logout is taken to 360 main store. The current PSW is written out to main store location 0. The 360 general registers and floating point registers are written to mainstore locations 256 through 280. (100 through 15C hex). Emulator status is written to locations 160 through 16B, and device registers to locations 16C through 17F. Finally, microstore is written out to main store locations C000 through FFFF (the upper 4k words). The CPU then halts.

5.2 Restart

A restart facility is provided, where the PSW is loaded from location 0 (Restart New PSW) and the registers restored from locations 100 through 15F (hex). Devices registers are cleared to zero (channel reset). Processing continues.

Logout/Restart facility is provided to allow processing to be interrupted and resumed later. Errors can be corrected in the microcode and the 360 program restarted.

5.3 Initial Program Load

Initial Program Load is accomplished by microcode. A record is read from a device into mainstore (the microcode assumes that the control console has accomplished this). The double word at location zero becomes the new PSW and processing begins at the location specified.

Special functions are provided to clear main store and local store (registers) to zero. The processor halts following each of these operations. Running the processor from here initiates the IPL sequence.

360 Fixed Storage Locations

Address	Hex Address	Length	Function
0	0	8	Initial Program Load PSW - Restart New PSW
8	8	16	unused
24	18	8	External Old PSW
32	20	8	Supervisor Call Old PSW
40	28	8	Program Old PSW
48	30	8	Machine Check Old PSW
56	38	8	Input/Output Old PSW
64	40	8	Channel Status Word
72	48	8	unused
80	50	4	Interval Timer
84	54	4	unused
88	58	8	External New PSW
96	60	8	Supervisor Call New PSW
104	68	8	Program New PSW
112	70	8	Machine Check New PSW
120	78	8	Input/Output New PSW
128	80	128	unused
256	100	96	Diagnostic Logout Area
256	100	64	General Registers (0-16)
320	160	12	CPU Status
332	16C	20	EXternal, Multiplexor, and Selector Channel Status,Device Registers.

5.4 Invoking Special Functions

The special functions of the EMMY/360 processor can be invoked by forcing a trap to the start of the proper microcoded routine. The routine for invoking a trap from the maintainance console will differ with the type of programming support. The procedure outlined here applies to the Debug G program (binary hardware diagnostic).

5.4.1 Initiating Microroutines from the Maintainance Console

Before performing any function involving altering the microinstruction stream or processor registers, the machine must be halted. Otherwise, results will be unpredictable.

Halt the processor from the display console by depressing the HALT toggle switch, or type "HALT" on the Datapoint Keyboard. When the procesor has halted, type 028<return>. The 028 must appear under the hashmarks on the CRT. The console will now be displaying the Special Function Trap Vector. The vector should be zero in the high 20 bit positions. Type the address of the desired special function in the low 12 bits of the display (in binary at present). Terminate with <return>. Type T<return>. This will issue an interrupt to the displayed address and the processor will initiate the special function. If it is desired, bit 15 of the displayed trap vector may be set to 1. This will cause the processor to halt before beginning the invoked function.

Special Function Addresses

Microstore Address	Symbolic Name	Function
076	$CLEARLS	Clear local storage
072	$CLEARMS	Clear main storage
05F	$DUMP	Dump microstore to mainstore (high 4K words)
06E	$IPL	Initial Program Load
050	$LOGOT	Diagnostic Logout and $DUMP
064	$RESTART	Reset local store and PSW restar

6.0 Emulator Status and Instruction Interpretation

6.1 The Program Status Word

The 360 PSW is parsed and stored in decoded form. Since the PSW is accessible only through explicit reference, the PSW may be stored in a form convenient for the emulation.

When referenced, only the parts needed are assembled to 360 format. These references include conditional branching, branch and link, and interrupt generation.

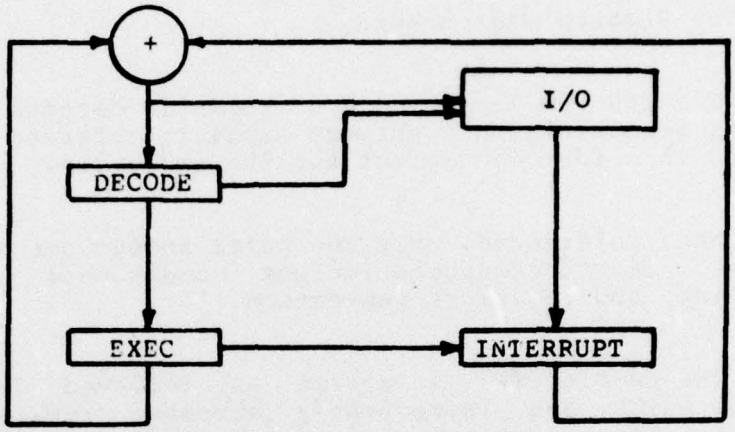
The 64-bit PSW is stored as recieved in Local Storage. Fields which are infrequently accessed remain stored in 360 format. These fields include System Mask (bits 63:56), Wait and Problem State (bits 49:48), and the Program Mask (bits 27:24), The Key field (bits 55:52) is ignored. The Interrupt Code field (bits 47:32) is cleared to zero prior to storage.

The Instruction Length Code as specified in the PSW is ignored. Each instruction cycle, a new ILC is obtained from the Semantic Pointer. The Condition Code is left justified and stored in Local Storage. During instructions which modify the 360 condition code, the micromachine control register, or a register with the required condition code in the high two bits, is stored in control store. The condition code is left justified, since the host MAR contains encoded CCODES in these bit positions. This minimizes target machine condition code modification overhead.

The next instruction address (NIA, PSW<23:0>) is placed in a host register (PC, register 1), concatenated on the left with an MMC byte specifying 4 byte transfer on a byte address (X'30'). All 24 bits of NIA are saved and used. No checking is done as to validity of the address, no boundary alignment restrictions are enforced, and no checking is done as to overflow into the MMC field of the PC upon incrementing of the NIA.

6.2 Interpretation Loop

Phases DECODE and EXEC comprise what is known as the interpretation loop of the emulation. DECODE and EXEC proceed until an interruption is generated (through external action or as the result of an exception or SVC instruction).



Interpretation Loop

Channel Emulator and Interrupt

6.2.1 Decode

Target instructions are decoded by a two level process. The Instruction Length Indicator (high 2 bits of the operation code) is used to select the proper format decode routine, and the entire operation code is used to select a Semantic Pointer to the execution semantic code.

Host Register Assignments

Name	Register	Purpose
MAR	R0	Micro Status Register
PC	R1	Target Program Counter
XR	R2	Scratch
IR	R3	Target Instruction Register
P	R4	holds Target I2 specification
Q	R5	holds Target Operand 2 address
R	R6	holds Target R1 specification
S	R7	holds Semantic Pointer

Upon entry to the decode routine, the XR (host register 2) is assumed cleared to all ones and the IR (host register 3) contains the next instruction to be interpreted. These registers are set by the previous execution routine. No checking is done as to the validity of these registers.

Interrupts are enabled by the first instruction of DECODE (during the execute phase, hard interrupts are disabled). An interrupt of the CPU by an external device causes this instruction to be replaced by a trap instruction to the proper interrupt generation code. In this way, it is ensured that the currently executing instruction will complete before the interrupt is taken.

The high 2 bits of the IR are shifted into the XR. This puts a negative number in the XR (-1 for SS instructions, -2 for SI instructions, -3 for RX instructions and -4 for RR instructions). This value is called the Format Index.

The rest of the operation code is then shifted into the XR, leaving the opcode right justified and one filled in XR, and the remaining bits of the instruction left justified in the IR.

The XR is used to address control store and select one of 256 semantic pointers from the high-order 256 words of control store.

Subtracting the format index from the MAR causes a branch foreward of 1, 2, 3, or 4 words beyond the current value of the MAR. (For SS, SI, RX, and RR formats respectively). The SS, SI, and RX decode routines are entered via a branch table of 3 consecutive words, while the RR decode lies in-line following the branch table. The various format decode routines complete parsing of the instruction and interpretation of the fields as outlined in the Principles of Operation for the System/360 (reference 2). Effective addresses are calculated where necessary.

The parsed instruction is passed to the execute semantic routine in predefined registers. The values contained in various registers are determined by the particular operation. The PC (next instruction address) is updated in some cases by the decode routine, and in others must be done by the semantic routine.

RR R-register (host R6) contains R1 specification (right justified, zero filled.
XR contains R1 specification
IR contains R2 specification (left justified)

Semantic code must parse R2 specification and update the PC (increment by 2).

RX R contains the R1 specification
Q (host R5) contains the operand 2 storage address as a 24 bit value. Bits 31:24 are unpredictable.
S (host R7) contains the Semantic Pointer, with MMC in bits 31:24.

The semantic code must update the PC (increment by 4).

SI,RS R contains R1 specification (instruction bits 23:20)
P (host R4) contains the I1 specification (instruction bits 23:16)
Q contains the 24 bit operand 2 storage address
S contains the semantic pointer

The PC is updated by the decode routine, and should not be modified.

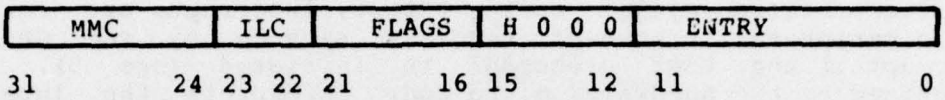
SS R contains the length specification(s) (instruction bits 23:16)
IR contains operand 1 address (24 bits, the high 8 bits are unpredictable)
Q contains operand 2 storage address.
S contains the semantic pointer

The PC is updated by the decode routine and should not be modified.

6.2.2 The Semantic Pointer

The upper 256 words of control store contain a table of semantic pointers - one for each possible 360 operation code. Each semantic pointer contains information which controls the interpretation of a particular instruction.

Semantic Pointer



MMC - memory controller command byte (storage operations only)
ILC - actual Instruction Length Code for this operation
FLAGS - operation dependent flags initially ones
H - halt indicator - normally zero, 1 indicates "halt when semantic code entered".
reserved - must be zeros
ENTRY - address of the semantic code for this operation

6.2.3 Execution Semantics

The execution semantic code performs the required target operation and sets up the registers for the next decode. Setup requires the prefetch of the next instruction and clearing of XR to ones.

The majority of 360 operation codes are invalid and generate Operation Exception interruptions. Three operation exception semantics are included. One each for RR and RX, where the PC must be updated before the interruption can be formatted. A third is provided for the SI and SS classes of instructions, where the interrupt may be formatted immediately.

Details of each semantic routine are not included.

6.3 Interruptions of the 360 Program

Interrupts of the 360 program may result from three distinct actions - external action (EXTERNAL and I/O interrupts), programmed action (SVC instruction), and exceptional conditions (PROGRAM). The actual PSW Swap is handled by a single routine. Interrupts are expected infrequently, thus the extra overhead of formatting a call to one swap routine is acceptable.

6.3.1 External Action

Hardware interrupts are disabled during the execute phase of the interpretation cycle. During DECODE, interrupts are enabled. When a peripheral processor requires service by the CPU, an interrupt of the EMMY processor is initiated (see 3). When recognized by the hardware, micro code interprets the interrupt and determines is the interrupt is to be passed on to the program (ie whether the interrupt is masked). If allowed by the SYSTEM MASK of the PSW, the first instruction of the decode loop is replaced by a trap instruction- a branch to the proper micro code to format the interrupt. Hardware interrupts are disabled. Execution resumes. In this way, it is guaranteed that the currently executing instruction completes execution before the interrupt is serviced (that is, the interrupt will occur between 360 instructions). If the interrupt is masked out (mask bit is zero), the channel status is saved, channel flagged as being in the "interrupt pending" state, and processing resumes. No trap instruction is inserted and hardware interrupts remain enabled. No interrupt of the 360 program is generated.

The program will be informed of the Interrupt Pending state the next time the channel is accessed (through an I/O Function Call). Channel Status will be stored in the Channel Status Word of main store (CSW) and condition code set. Note that this differs from 360 in that, when an interrupt is masked, it remains pending and will occur as soon as the system mask allows. In EMMY/360, once channel status has been stored in the Channel Status register of local store and the channel flagged as interrupt pending, no further interrupt action will occur. A more appropriate flag for the emulated channel would be "Interrupt Suppressed".

The 360 Start I/O and Halt I/O instructions are invalid and will result in operation exception conditions. Test I/O will return the same condition codes as its 360 counterpart, as well as clear the "Interrupt Pending" state.

External and I/O interrupts are formatted by retrieving the appropriate device (or interrupt class) and passing this, along with Old PSW address as a (32-bit) word address, rather than a byte address, to the PSW Swap routine.

6.3.2 Supervisor Call

Supervisor Call (SVC) causes a call to PSW Swap to be formatted. The R1-R2 field of the instruction is passed as interrupt code.

6.3.3 Exceptional Conditions

Two types of exceptional conditions may arise during execution of an operation. One type causes an interrupt following execution of the instruction and is intended to warn of a potentially dangerous result being detected during execution. The second prohibits execution of the instruction (for reasons such as invalid data) and causes the operation to be suppressed and an interrupt to be generated immediately. In this case, no operands are modified.

When a prohibitive condition is detected, a PSW swap is immediately formatted and execution semantic code exited. When a questionable condition is detected, a note of the condition is made and execution completed. Results are stored. If any Program Mask bits are applicable, these are interrogated and, if set, an interrupt formatted upon completion of the operation.

6.3.4 PSW Swap

A single routine is responsible for PSW swap. This includes the assembly of the current PSW into 360 format, inserting the specified interrupt code in bits 47:32, and storing at the specified Old PSW location. A new PSW is fetched from Old PSW plus 16 and decoded.

The Q register contains the Old PSW location, as a word address. The new PSW is always retrieved from this address plus 16. XR contains the interrupt code to be inserted in bits 47:32 of the old PSW. This value is simply added to the high 32 bits of the Current PSW prior to storage in main store, thus the low 16 bits of the high word of the current PSW are cleared upon loading a new PSW, and the high 16 bits of XR must be zero upon entry to PSW swap.

The ILC is obtained from MAR<23:22>. The NIA is obtained from the PC (host register 1) bits 23:0. The condition code is obtained from control store, shifted right 2 bits for alignment, and inserted in PSW<29:28>. For Branch and Link instructions, only the low 32 bits of the current PSW are formatted, and no new PSW is processed.

The new PSW is retrieved and decoded. Various fields are stored at the appropriate locations. A display status word is prepared with Wait and (inverted) Problem state bits in bits 31:30 and the specified NIA in the low 24 bits. This is displayed on

the console and stored in control store. If the wait state bit is set, a wait loop is entered with hardware interrupts enabled. The interrupt bit of the MAR is tested each loop and, if reset, decode resumed. If an interrupt occurred during the wait loop, the interrupt bit will have been reset and a trap inserted in the decode loop. Thus, the wait loop will be terminated and the interrupt processed.

If the wait state was not specified, the XR is cleared to ones and the specified next instruction fetched. Decode resumes.

7.0 Status of EMMY/360 Project

Code for the interpretation loop has been written and the decode routines tested. Semantic routines have also been checked, though not to the extent the decode routines have. A 360 program was written and used to test the functionality of the various 360 operations.

Approximately 1400 words of EMMY Control Store are occupied by emulator code. Semantic pointers occupy an additional 256 words. This leaves approximately 2400 words free for the implementation of additional instructions and I/O support.

I/O support must still be developed for the various peripheral processors. I/O interrupt generation is included in the emulator; only the semantic code for the actual I/O Function Call instruction need be written.

Floating point semantics must be added to the emulator. This code has already been developed for DELTRAN [10], and need only be copied, with code to test for 360 exceptional conditions.

Streams of 360 instructions have been executed using the emulator. The DECODE/EXEC interpretation loop performed reliably,though some hardware problems were encountered. Due to hardware availability, instruction timing determination is incomplete.

ACKNOWLEDGEMENT

Many people contributed to the development of the EMMY/360 and it is not possible to mention all here. However, the author wishes to thank, in particular, Lee Hoevel, Robert McClure, and Charles Neuhauser for their comments, suggestions, and contributions to the design of the EMMY/360.

References

1. Datapoint Corp., Datapoint 200 Reference Manual, Datapoint Corp., 9725 Datapoint Drive, San Antonio, Texas 78284.

2. IBM Corp., System/360 Principles of Operation, order no GA22-6821-8.

3. Neuhauser, C., An Emulation Oriented. Dynamic Microprogrammable Processor (Version 3),TN 65, October 1975, Digital Systems Lab, Stanford University, Stanford, Ca 94305.

4. Neuhauser, C., EMMY System Peripherals -- Principles of Operation, TN 77, December 1975, Digital Systems Lab, Stanford University, Stanford, Ca. 94305.

5. Hoevel, L. and Wallach, W., A Tale of Three Emulators, TR 98, October 1975, Digital Systems Lab, Stanford University, Stanford, Ca 94305.

6. Wallach, W., 360 Emulator Performance Estimate, TN 66, October 1975, Digital Systems Lab, Stanford University, Stanford Ca. 94305.

7. Wallach, W., EMMY/Unibus Interface-Preliminary Specification, TN 88, June 1976, Digital Systems Lab, Stanford University, Stanford Ca., 94305.

8. Wallach, W., Virtual Addressing for EMMY/360, TN 89, June 1976, Digital Systems Lab, Stanford University, Stanford Ca. 94305.

9. Wirth, N., The PL360 System, TR CS91, April 1968, Computer Science Department, Stanford University, Stanford Ca., 94305.

10. Hoevel, L., report on DELTRAN (direct execution of FORTRAN) to be issued

Appendix A - Notes on I/O Support

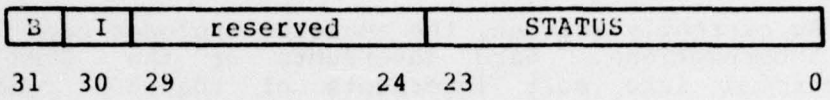
As currently written, the emulator includes code to support I/O interruptions. Hard interrupts of the EMMY CPU are interpreted into soft interrupts of the 360 program. The peripheral processor must supply certain information.

Each potential source of hard interrupt (that is, sources capable of producing a 360 EXTERNAL or I/O interrupt) is provided with a Device Register. I/O channel processors are also provided with a Status Register. Just prior to initiating an interruption of the EMMY CPU, the peripheral processor should write the device identifier (channel/device for I/O, EXTERNAL SOURCE for EXTERNAL) to its Device Register. The EXTERNAL Device Register is located at Control Store location X'1B', Selector Channel 0 Device Register (the Datapoint) at location X'1D', and Selector Channel 1 (not installed) at location X'1F'. The low 16 bits of this register will be inserted into the Interrupt Code field of the Old PSW upon interrupt generation (the high 16 bits must be zero, or the high 16 bits of the Old PSW will be unpredictable).

The I/O processors are also supplied with a Status Register. The CPU should set the high bit of this register when an operation involving its associated channel is initiated, indicating the channel is busy. When the peripheral processor completes the I/O operation, this bit should be cleared. Any relevant status information should be placed in the low 24 bits of the register. This information will be supplied to the 360 program upon interrupt.

If a peripheral processor attempts to interrupt the 360 program while its mask bit is zero, bit 30 of the channel's status register will be set by the CPU. This indicates there is an interrupt pending for that channel and inhibits further access of that channel until the 360 program clears the interrupt. The interrupt may be cleared by issuing a Test I/O instruction or an I/O Function Call. In each case, the Channel Status Register will be stored in main store (location 64, the Channel Status Word) and condition code set to 01.

Channel Status Register



B	BUSY-1 indicates channel is busy
I	Interrupt Pending-1 indicates an interrupt from this channel is pending
STATUS	Channel and Device status bits dependent upon channel processor and device characteristics

List of Instructions by Set and Feature

Standard Instruction Set

NAME	MNEMONIC	TYPE		CODE
Add	AR	RR	C	1A
Add	A	RX	C	5A
Add Halfword	AH	RX	C	4A
Add Logical	ALR	RR	C	1E
Add Logical	AL	RX	C	5E
AND	NR	RR	C	14
AND	N	RX	C	54
AND	NI	SI	C	94
AND	NC	SS	C	D4
Branch and Link	BALR	RR		05
Branch and Link	BAL	RX		45
Branch on Condition	BCR	RR		07
Branch on Condition	BC	RX		47
Branch on Count	BCTR	RR		06
Branch on Count	BCT	RX		46
Branch on Index High	BXH	RS		86
Branch on Index Low or Equal	BXLE	RS		87
Compare	CR	RR	C	19
Compare	C	RX	C	59
Compare Halfword	CH	RX	C	49
Compare Logical	CLR	RR	C	15
Compare Logical	CL	RX	C	55
Compare Logical	CLC	SS	C	D5
Compare Logical	CLI	SI	C	95
Convert to Binary	CVB	RX		4F
Convert to Decimal	CVD	RX		4E
Divide	DR	RR		1D
Divide	D	RX		5D
Exclusive OR	XR	RR	C	17
Exclusive OR	X	RX	C	57
Exclusive OR	XI	SI	C	97
Exclusive OR	XC	SS	C	D7
Execute	EX	RX		44
Insert Character	IC	RX		43
Load	LR	RR		18
Load	L	RX		58
Load Address	LA	RX		41
Load and Test	LTR	RR	C	12
Load Complement	LCR	RR	C	13
Load Halfword	LH	RX		48
Load Multiple	LM	RS		98
Load Negative	LNR	RR	C	11
Load Positive	LPR	RR	C	10
Load PSW	LPSW	SI	L	82
Move	MVI	SI		92
Move	MVC	SS		D2
Move Numerics	MVN	SS		D1
Move with Offset	MVO	SS		F1
Move Zones	MVZ	SS		D3
Multiply	MR	RR		1C
Multiply	M	RX		5C
Multiply Halfword	MH	RX		4C
OR	OR	RR	C	16
OR	O	RX	C	56
OR	OI	SI	C	96
OR	OC	SS	C	D6
Pack	PACK	SS		F2
Set Program Mask	SPM	RR	L	04
Set System Mask	SSM	SI		80
Shift Left Double	SLDA	RS	C	8F
Shift Left Single	SLA	RS	C	8B
Shift Left Double Logical	SLDL	RS		8D
Shift Left Single Logical	SLL	RS		89
Shift Right Double	SRDA	RS	C	8E
Shift Right Single	SRA	RS	C	8A
Shift Right Double Logical	SRDL	RS		8C
Shift Right Single Logical	SRL	RS		88
Store	ST	RX		50
Store Character	STC	RX		42
Store Halfword	STH	RX		40
Store Multiple	STM	RS		90
Subtract	SR	RR	C	1B
Subtract	S	RX	C	5B
Subtract Halfword	SH	RX	C	4B
Subtract Logical	SLR	RR	C	1F
Subtract Logical	SL	RX	C	5F
Supervisor Call	SVC	RR		0A
Test and Set	TS	SI	C	93
Test Under Mask	TM	SI	C	91
Translate	*TR*	*SS*		DC
Translate and Test	TRT	SS	C	DD
Unpack	UNPK	SS		F3

Key

C Condition Code set
L new Condition Code loaded

(Reproduced from Reference 2)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ~~Technical Report # 114~~ 12958.10-M	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EMMY/360 FUNCTIONAL CHARACTERISTICS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER SEL-76-024
7. AUTHOR(s) Walter A. Wallach		8. CONTRACT OR GRANT NUMBER(s) DAAG29-76-G-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Digital Systems Laboratory Stanford Electronics Laboratories Stanford University, Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office-Durham		12. REPORT DATE June 1976
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An emulation of the IBM System/360 architecture is presented for the EMMY/360. Problem state code which executes correctly on an IBM 360 will also execute correctly on the EMMY/360. Code producing execution exceptions will, in most cases, produce the same results on the two systems. Certain exceptions occurring on the IBM 360 cannot occur on the EMMY/360, such as address specification exceptions for main store operands, and certain precise interrupts on IBM 360 will be imprecise on the EMMY/360, such as address exceptions. The EMMY/360 supports

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

the Stanford 360 instruction set with single precision floating point. The 360 input/output structure is not supported; I/O on the EMMY system is done by Function Call instruction, rather than channel program and Start-Test I/O.